Optimal design of switched Ethernet networks implementing the Multiple Spanning Tree Protocol

Bernard Fortz[a], Luís Gouveia[b], Martim Joyce-Moniz[a,*]

[a] *Université Libre de Bruxelles, Département d'Informatique, CP 212, Boulevard du Triomphe, 1050 Bruxelles, Belgium and INOCS, INRIA Lille Nord-Europe, France*
[b] *Departamento de Estatística e Investigação Operacional, Faculdade de Ciências da Universidade de Lisboa, Bloco C / 2 - Campo Grande, Cidade Universitária, 1700 Lisboa, Portugal*

Abstract

Switched Ethernet networks rely on the Spanning Tree Protocol (STP) to ensure a cycle-free connectivity between nodes, by reducing the topology of the network to a spanning tree. The Multiple Spanning Tree Protocol (MSTP) allows for the providers to partition the traffic in the network and assign it to different virtual local area networks, each satisfying the STP. In this manner, it is possible to make a more efficient use of the physical resources in the network. In this paper we consider the traffic engineering problem of finding optimal designs of switched Ethernet networks implementing the MSTP, such that the worst-case link utilization is minimized. We show that this problem is $\mathcal{NP}$-hard. We propose three mixed-integer linear programming formulations for this problem. Through a large set of computational experiments, we compare the performance of these formulations. Until now, the problem was almost exclusively solved with heuristics. Our objective here is provide a first comparison of different models that can be used in exact methods.

Keywords: Telecommunications, Traffic Engineering, Multiple Spanning Tree Protocol, Network Design, Mixed-Integer Programming

*Corresponding author
Email addresses: bernard.fortz@ulb.ac.be (Bernard Fortz), legouveia@fc.ul.pt (Luís Gouveia), martim.moniz@ulb.ac.be (Martim Joyce-Moniz)

1. Introduction

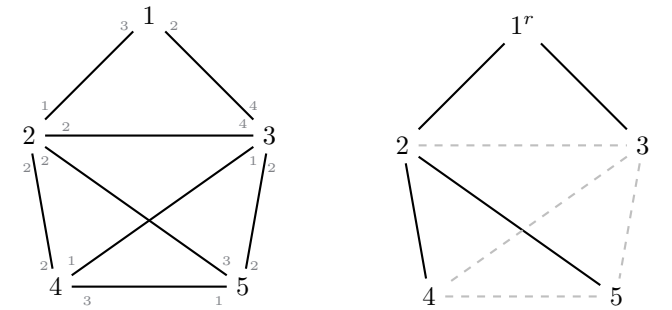
The field of traffic engineering was originally characterized by the application of probability theory to telecommunications networks. The array of tools used in the discipline included stochastic processes, queueing theory and numerical simulation, and the goal was to evaluate, operate and maintain these networks [1]. Meanwhile, the field has increased in importance, as well as in breadth. Nowadays, traffic engineering also includes a wide array of optimization methods, that aim at finding the best network configuration, to improve the quality of service (QoS) [2].

In this paper, we consider a traffic engineering problem that arises while designing data center networks. Data centers play a crucial role in today's society, by housing huge numbers of servers, responsible for hosting the increasingly popular Internet and cloud computing services. In these data centers, switched Ethernet networks are becoming a common choice, as they offer better port density at a lower price per Gbps (billions of bits per second). Better port density translates in the capacity to carry larger amounts of traffic flow per unit of space in the data center.

In switched Ethernet networks, it is important to keep redundant links to ensure automatic backup paths in case of link failure. However, the existence of cycles in these network topologies can result in broadcast storms, *i.e.* the accumulation of broadcast and multicast traffic. This happens because the switches in the cycle repeatedly rebroadcast the data packets, flooding the network [3]. Ultimately, broadcast radiation can have a high impact on the performance of the network and should, therefore, be avoided at all costs. As such, switched Ethernet networks only activate, at a given time, a cycle-free subset of the existing links.

In this sense, these networks implement the Institute of Electrical and Electronics Engineers (IEEE) 802.1d standard [4], also known as the Spanning Tree Protocol. As the name of the protocol indicates, the topology of the subset of activated links, in a network using 802.1d, must be a spanning tree.

To configure the spanning trees, each switch is assigned two types of integer values: a BridgeID and a PortCost for every of its ports. The switch with the lowest BridgeID is chosen to be the Root Bridge. Then, the actived links are deduced from the PortCosts: a link is selected, if it is part of the minimum cost path between each switch and the Root Bridge. The cost of each path is calculated by summing the PortCosts of the forwarding ports. If there exists two paths with the minimum cost, between a given switch and the Root Bridge, the BridgeID of the second switch in the path is used to break the tie. An example for this procedure is depicted in Figure 1.



(a) Assignment of PortCosts (in gray) and BridgeIDs (same value as node id).

(b) Activated spanning tree, highlighted in black.

Figure 1: Example of PortCost and BridgeID assignment and resulting spanning tree. Node 1 is chosen as the Root Bridge. Even though path {1,2,4} and {1,3,4} have both length 8, the first is chosen because the BridgeID of node 2 is smaller than of node 4.

One of the drawbacks of this protocol is that the network ends up only using a small number of existing links.

The IEEE 802.1q standard [5] enables large switched Ethernet networks to be partitioned in multiple, smaller virtual local area networks (VLANs), simplifying the design of the network. This allows for the isolation of different applications and/or data center customers, as two nodes belonging to a given VLAN can only communicate between each other, through the links established for the same VLAN.

The Multiple Spanning Tree Protocol (MSTP), standardized as 802.1s [6],

allows for service providers to install different spanning trees (one per VLAN) over a single physical topology. This is highly advantageous for the traffic performances of switched Ethernet networks, as the traffic can be spread throughout a bigger number of links. Nevertheless, the MSTP is hard to use effectively, since optimizing over multiple spanning trees in one single network is a highly combinatorial problem. In practice, current implementations of the MSTP circumvent this, by computing a small number of spanning trees, and mapping the VLAN onto them. Often, when many VLAN are defined, a spanning tree is generated, not for each single VLAN, but for a whole subset of them [7]. In all likelihood, these implementation result in designs that do not make the best use of the network resources. Many methods have been proposed in the literature to improve the implementation of the MSTP, with respect to different traffic-oriented measures. Our objective here is to study the optimization problem arising from the use of the MSTP protocol.

Extensions to this protocol, better suited for traffic management, have been proposed in the engineering literature (see e.g. [8, 9]), but these are outside the scope of this paper and could be a topic for further research. Nevertheless, we think the MSTPtechnology is worth studying as it is in use in many networks today, and it raises challenging problems for optimizers.

Other technologies like Software Defined Networks (SDNs) might be better suited for large data centers. However, SDNs are much more complex to manage and the commercially available solutions are expensive and quite limited in their traffic engineering capabilities. On the other hand, switched Ethernet equipment is a mature technology, it is quite inexpensive and its management requires less expertise human resources (which also has an impact on the operational costs).

The literature review in Section 4 shows that the problem of traffic engineering in MSTP operated networks was almost exclusively tackled with heuristic methods. These methods are reported to be able to efficiently produce good designs, with respect to different performance measures. Notwithstanding, little is known about the optimality or quality of these solutions, as dual bounds to the optimum values of the respective problems are seldom provided. Accordingly, in

this paper, we propose mathematical programming formulations for a network design problem dealing with the MSTP, that can yield optimal designs, or at the very least, produce bounds that can be used to shed light on the quality of heuristic solutions.

We focus on the problem of finding optimal designs for switched Ethernet networks implementing MSTP, first studied by Ho *et al.* [10, 7]. We denote it as the traffic engineering for MSTP problem (TE-MSTPP). This problem consists in designing networks with multiple VLANs, such that each VLAN is defined by a spanning tree in which traffic demands can be routed without exceeding the link capacities. The traffic-oriented performance measure that we consider, is the minimization of the worst-case link utilization in the network. The utilization of a link is defined as the ratio between its load (*i.e.* the sum of traffic flowing on it) and its bandwidth. Therefore, if the utilization of a link exceeds 1, a link is considered to be overloaded.

In this paper, we do not consider the assignment of the BridgeIDs and Port-Costs to the switches. It is possible to do this *a posteriori*, such that the spanning trees selected in the optimization are implemented by the MSTP [11] .

Note that we are not the first to consider mathematical programming for problems dealing with the MSTP. In fact, as described in Section 4, other works have previously proposed mathematical programming formulations for such problems, some possibly more complex than the TE-MSTPP. Nevertheless, to the best of our knowledge, no extensive study has been made, that analyses and compares different formulations, identifying their strengths and weaknesses. For instance, in these works we did not find a single case where the bounds obtained by the linear programming (LP) relaxation of the proposed formulations are reported. We believe that such a study is of great relevance, and fills an important gap in the current state of the art. Furthermore, we believe that the TE-MSTPP has basic, representative characteristics, that can be found in most problems regarding the MSTP; as such, this problem is an obvious choice for the analysis we propose.

In Section 2, we introduce some concepts of graph theory and combinatorial optimization, that will be used throughout this paper. In Section 3, we define the TE-MSTPP. The TE-MSTPP is shown to be $\mathcal{NP}$-hard in Section 5. Then, we propose three different MILP formulations to model the TE-MSTPP, in Section 6. In Section 7 we compare the LP relaxation of these formulations. Computational experiments are presented in Section 8, to further compare the proposed formulations. Finally, in Section 9, we draw some conclusions and discuss future developments.

2. Notation and definitions

A switched Ethernet network can be represented by a graph, where nodes typically represent computers or switches, and edges represent the bi-directional links connecting the latter. In the section, we introduce some concepts of graph theory and combinatorial optimization, that will be used throughout this paper.

Consider an undirected graph $G = (V, E)$, where V is the set of nodes, with size n, and E the set of edges, with size m. Edge $e = \{i, j\} \in E$ represents an undirected link between the two end nodes, $i \in V$ and $j \in V$. Given a set $W \subset V$, $\delta(W) = \{\{i, j\} \in E : i \in W, j \in V \backslash W\}$ denotes the cut induced by W. The degree of a node v is defined as the cardinality of $\delta(v)$.

A sequence of edges in E connecting two nodes in V is called a path. A cycle is defined as a path, that starts and ends at the same node. A graph that has a path between each pair of nodes is a connected graph.

Consider as well the set of arcs $A = \{(i, j), (j, i) : \{i, j\} \in E\}$. The graph $G' = (V, A)$ is the *directed* version of graph G. For such graph, and for a given set $W \subset V$, we can define the two following cuts: $\delta^-(W) = \{(i, j) \in A : i \in V \backslash W, j \in W\}$ and $\delta^+(W) = \{(j, i) \in A : i \in V \backslash W, j \in W\}$. The cardinality of $\delta^-(v)$ and $\delta^+(v)$ is respectively named as the indegree and outdegree of node v.

For simplicity, in many situations, we use the same designation for concepts defined in the directed graph as for the equivalent concepts defined in the in undirected graph, *e.g.* "paths", "cycles". This is not the case, however, for

spanning trees and arborescences. A spanning tree of G is a connected subgraph, that includes all the nodes in V and contains exactly $n-1$ edges. Consequently, a spanning tree is also acyclic, in the sense that it contains no cycles. An r-arborescence is a subset of A, such that there is no arc entering the root node r, and there is an unique path between r and every other node in V.

As stated in the previous section, in this paper we propose several Mixed Integer Linear Programming (MILP) formulations. Consider the LP relaxation of a MILP formulation F, where the integrality of the variables is relaxed, and which we denote by F_{LP}. $B_{LP}(F)$ stands for the bound provided by solving F_{LP}. We also denote $\mathcal{P}_F$ as the polyhedron defined by the set of feasible solutions of F_{LP}. The strength of the LP relaxation of two different MILP formulations, F^1 and F^2, can then be compared by examining their respective polyhedra, $\mathcal{P}_{F^1}$ and $\mathcal{P}_{F^2}$. The LP relaxation of F^1 is as strong as (strictly stronger) F_2 if $\mathcal{P}_{F^1} \subseteq \mathcal{P}_{F^2}$ ($\mathcal{P}_{F^1} \subset \mathcal{P}_{F^2}$). They are regarded as equivalent, if $\mathcal{P}_{F^1} = \mathcal{P}_{F^2}$.

We consider as well the concept of projection, which provides a connection between different formulations. Given a polyhedron $Q = \{(u, x) \in \mathbb{R}^p \times \mathbb{R}^q : Au + Bx \leq b\}$, the projection of Q onto $\mathbb{R}^q$, or onto the x-space, is defined as $Proj_x(Q) = \{x \in \mathbb{R}^q : \exists u \in \mathbb{R}^p : (u, x) \in Q\}$. This is useful as it allows us to compare formulations in different variable spaces.

3. Problem definition

In this section, we define the TE-MSTPP. Let $G = (V, E)$ be an undirected graph, as defined in the previous section. Consider that in G, each edge $e = \{i, j\} \in E$ is assigned a capacity, denoted by C_e. This capacity is regarded as symmetric, in the sense that it limits the sum of traffic flowing in both directions, (i, j) and (j, i), together.

We also define S as the set of VLANs in the switched Ethernet network. For each VLAN $s \in S$, $d_s(u, v)$ represents the traffic demand between nodes $u \in V$ and $v \in V$. We assume for simplicity, that u must be smaller than v, and $d_s(u, v)$ stands for the sum of traffic to be sent, both from u to v and from v to

u.

The TE-MSTPP consists of finding a design of all VLANs $s \in S$, such that we minimize the worst-case link utilization; the ratio between the total load on the link and its capacity. Furthermore, a feasible set of VLANs must satisfy the following properties:

- the topology of each VLAN is a spanning tree;
- all given traffic demands in a VLAN are routed;
- the total traffic flowing through a link does not exceed its given capacity.

4. State of the art

Ho [7] wrote an extensive review on several approaches from the literature, regarding traffic engineering problems for Ethernet networks implementing the MSTP.

In a first approach, [12, 13, 14] proposed optimization techniques that map a set of VLANs to a given number of spanning trees. Meddeb [14] developed an algorithm to generate a set of spanning trees with a small number of links in common, and then introduced another greedy algorithm to map each VLAN to those spanning trees, while attempting to minimize the number of used links. Lim *et al.* [13] proposed a QoS-aware mechanism that maps VLANs, with the objective of minimizing network load and delay. He *et al.* [12] used an admission control algorithm to assign a group of VLANs to each given spanning tree, and then map each service to a VLAN, such that it minimizes the link load.

Sousa *et al.* [11] and Santos *et al.* [15, 16], introduced matheuristic schemes, that aim to balance the load in networks using the MSTP. In [15, 16], the heuristic procedure solves relaxed MILPs, in order to obtain feasible solutions and lower bounds. Different criteria were taken into consideration, including service disruption and network load balancing.

Chen *et al.* [17] proposed an algorithm that designs a spanning tree for every source node in the network, while trying to achieve a good trade off between

load balance and average delay.

One last approach was suggested by Padmaraj *et al.* [18] and Mirjalily *et al.* [19]. In this approach, costs are assigned to the links in the network, and the "cheapest" spanning trees are selected. In the first paper, the proposed heuristic updates weights assigned to the links in the network, in order to find a set of spanning trees with a good load balancing. In the second one, the suggested algorithm tries to find the best set of edge-disjoint spanning trees, and the best mapping of VLANs to that set.

Ho [7] argumented that all these proposals were not applicable for large networks. Hence, he proposed a local search based algorithm that aims at solving the TE-MSTPP.

Next, we review works that have modelled their respective problems as MILP. First, in [20], Ethernet networks using the MSTP are optimized with respect to QoS, while ensuring network protection, in case of link failure. In order to do so, the authors define, for each traffic demand, two spanning trees: a working tree and a protection tree. The set of traffic demands is partitioned in four classes, with different QoS priorities and bandwidth volume requirements. The objective is to minimize a weighted sum of the chosen resources in the network. In the proposed MILP, the unique path between the origin and destination of each traffic demand is formulated as a multicommodity flow (see Section 6). Moreover, the tree design variables are defined in the directed graph. In [21] a very similar model is presented, without the concern for tree protection. The authors do not seem to be so much focused on optimizing large-sized instances, as on exploring the impact of different protection and design strategies on the QoS of toy networks.

In [22], the authors propose an energy management framework for networks using the MSTP. They propose three MILP for three different problems: one that focus in optimizing the energy consumption in the network, one that considers load balancing objectives as in [15, 16], and one that integrates the two objectives into one single problem. In their work, [22] do not include the design of spanning trees in the MILP. Instead, they generate *a priori* and heuristically

a set of spanning trees, that can later be mapped to VLANs by the MILPs. In their computational experiments, they set the cardinality of this set as 30. Even for the smaller networks in their tests, with 17 nodes and 26 links, this means that a very large majority of potential designs are left out of the MILP optimization procedure. As such, even if this strategy might allow for a faster solving of the MILPs, it can also significantly cripple the quality of the final solutions.

Recently, [23] have proposed another protection scheme for the MSTP, with the purpose of quickly configuring a back-up VLAN in case of link failure. The authors require that the switches, adjacent to a given protected link, be leaf nodes if this link fails. The authors propose a MILP formulation for the problem of designing load-balanced and resilient Ethernet networks implementing the MSTP. In this formulation, the authors suggest transforming the original graph into a two-layered graph, such that the load-balanced working trees are set on the top layer, and the back-up trees on the bottom one. A back-up tree is selected for all failure scenarios, following the aforementioned protection scheme. In the proposed MILP, the design of the spanning trees is also not considered; they use instead a mapping strategy as the one described above, for [22]. Lee *et al.* suggest that their proposed MILP is to large to solve. Therefore, they partition the problem in two phases: in the first phase, the heuristic algorithm proposed in [11] is used to design the load-balanced working trees; in the second phase, given the set of working trees, the original MILP can be decomposed into smaller ones, where the protection for each failure scenario is ensured.

All the problems described above, despite dealing with the MSTP, clearly differ from the one considered in this paper. Although they are also distinct, the problems in [15, 16] are closer to the TE-MSTPP. In both papers, the same formulation is proposed, but different objectives are regarded. In [16] two load balancing problems are tackled: the minimization of the average link load, with a guaranteed optimal worst-case link load; and the minimization of the worst-case link load, with a guaranteed average link load. Moreover, in both [15] and [16] another problem is discussed, dealing with a lexicographical

minimization of the links' load. In this lexicographical objective, the authors begin to minimize the worst-case link load. Afterwards, that value is fixed, and the second largest load is minimized. The procedure continues in this fashion for all links. One other important distinction between the problems proposed in these papers and the TE-MSTPP, is that the assignment of each traffic demand is not given as an input, but rather regarded as a decision to be considered in the optimization procedure. The formulation used in both papers makes use of two sets of variables, in order to define the paths between the origin and destination of each traffic demand: one of multicommodity flow variables and one of rooted directed variables (see Section 6); and two sets of design variables to define each tree: one directed and one undirected.

Although both [15, 16] propose MILPs for their problem, the focus of these works is clearly on heuristic procedures. In fact, Santos *et al.* claim, as Cinkler *et al.* also had before, that the computational cost of using exact methods for these problems motivate the need for heuristic methods. In this sense, in [15, 16], the proposed MILPs are relaxed and embedded in a heuristic procedure. Although the authors also suggest exact methods *per se*, these seem to be used only as a means of measuring the quality of the proposed heuristics.

This review implies that, when it comes to the use of exact methods for the solving of problems dealing with the MSTP, there is still plenty of room for improvement, namely through the development of more efficient and strong formulations.

5. Problem complexity

In this section, we show that the TE-MSTPP is $\mathcal{NP}$-hard. A similar problem, the Optimum Communication Spanning Tree Problem [24], where traffic demands have to be routed over a single spanning tree, such that the communication cost is minimized, was proved to be $\mathcal{NP}$-hard. However, as far as we aware, there are no proofs of complexity for such a problem that is also capacitated.

We begin our proof by considering the corresponding decision problem (TE-MSTPDP), which is defined as follows: Let $G = (V, E)$ be an undirected graph, with capacity C_e assigned to each edge $e \in E$; and S a set of VLANs, where in each $s \in S$, we have traffic demands, $d_s(u, v)$, between nodes $u \in V$ and $v \in V$. TE-MSTPDP asks the question, "Is it possible to design each VLAN as a spanning tree, such that all the traffic demands are routed without exceeding the capacities on the link?".

We use the Boolean Satisfiability Problem (SAT) to demonstrate that this decision problem is $\mathcal{NP}$-complete. Then we conclude the proof by showing that if the TE-MSTPDP is $\mathcal{NP}$-complete, the TE-MSTPP is $\mathcal{NP}$-hard .

Theorem 1. *TE-MSTPDP is $\mathcal{NP}$-complete, even in the case where $|S| = 1$.*

Proof. Let (X, C) be an instance of SAT. $X = \{x_1, x_2, ..., x_\nu\}$ is the set of boolean variables, where each $x_i \in X$ can be either a positive literal (denoted by χ_i) or a negative literal (denoted by $\overline{\chi}_i$). $C = \{c_1, c_2, ..., c_\mu\}$ stands for the set of clauses, where each clause c is a disjunction of literals. In this version of the problem, each clause can have any given number of literals, from 1 to ν. A truth assignment $\{x_1^*, x_2^*, ..., x_\nu^*\}$ is defined as an assignment of the boolean value *true* or *false* to each variable $x \in X$. A given clause is said to be satisfied under that truth assignment, if it contains the literal χ and $x^* := true$, or the literal $\overline{\chi}$ and $x^* = false$. The objective of SAT is to find out if there is a truth assignment such that all clauses in C are satisfied. Cook and Levin proved that SAT is $\mathcal{NP}$-complete [25, 26].

We show that the TE-MSTPDP is polynomially reducible to SAT, and therefore, it is $\mathcal{NP}$-complete too. Consider an instance of SAT (X, C), as defined above. We now describe how to construct an undirected graph $G = (V, E)$, to model this instance.

For each variable $x \in X$ we consider a triple of nodes denoted as x^0, x^T and x^F. The first node acts as a "representative" of the variable, whereas the latter two imply the choice of assignment - respectively, *true* or *false*. Moreover, we consider a node for each clause $c \in C$ and a root node, denoted by r.

Let $C_\chi = \{c \in C : \chi \text{ appears in } c\}$ and $C_{\overline{\chi}} = \{c \in C : \overline{\chi} \text{ appears in } c\}$. The set E is comprised of the following group of edges, for each variable $x \in X$:

- $\{r, x^0\}$, with capacity μ;
- $\{x^0, x^T\}$, with capacity $|C_\chi|$;
- $\{x^0, x^F\}$, with capacity $|C_{\overline{\chi}}|$;
- $\{x^T, x^F\}$, with capacity $\mu + 1$;
- $\{x^T, c\}, c \in C_\chi$, with capacity 1;
- $\{x^F, c\}, c \in C_{\overline{\chi}}$, with capacity 1;

Finally, we assume that there is the following set of demands: $d\{r, c\} = 1$, for every clause $c \in C$; and $d\{x^T, x^F\} = \mu + 1$, for every variable $x \in X$.

It is easy to see that G can be constructed in polynomial time. Figure 5 depicts what this graph looks like for a SAT instance with four variables, and the following five clauses: $(\chi_1 \vee \overline{\chi}_4)$, $(\overline{\chi}_1 \vee \overline{\chi}_2 \vee \chi_3)$, $(\chi_3 \vee \chi_4)$, $(\chi_2 \vee \chi_4)$ and $(\overline{\chi_3} \vee \overline{\chi_4})$. The thicker edges represent the following feasible solution: $x_1 = false$, $x_2 = true$, $x_3 = true$, $x_4 = false$.

We now show that there exists a feasible truth assignment for SAT, if and only if the answer for the TE-MSTPDP is positive for G and the aforementioned set of demands.

Consider the feasible truth assignment $\{x_1^*, x_2^*, ..., x_\nu^*\}$. This solution can be represented by a sub-graph $T^* = (V, E^*) : E^* \subseteq E$, that routes the demands described above, whilst not surpassing the capacities on the edges. The assignment of a given variable $x \in X$ to *true* or *false*, is expressed by selecting respectively the edge $\{x^0, x^T\}$ or $\{x^0, x^F\}$. Each clause $c \in C$ has to be satisfied by, at least, one literal. This is ensured by the existence of a path in E^*, between the corresponding node and the root, such that the demand $d(r, c)$ can be routed. The last edge on the path connects the clause node with the node that indicates which variable assignment ensures the satisfiability of the clause. Even if the clause is satisfied by more than one literal, we only include

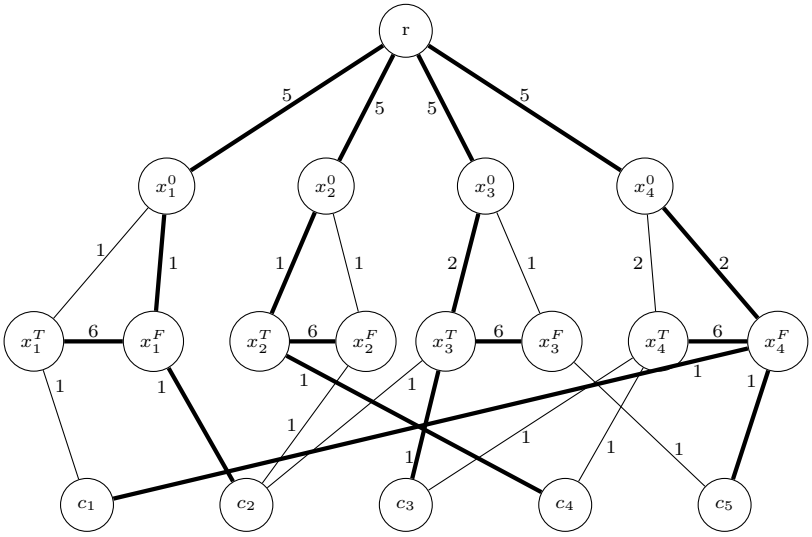


Figure 2: Graph construction for an example of a SAT instance.

one of these edges in E^*; the others are redundant. Finally, $\{x^T, x^F\} \in E^*$, as the direct edge is the only path through which we can send the corresponding demand, without exceeding the capacity. If the case arises that the sub-graph constructed in this fashion is not connected (*e.g.* if there are more variables than clauses), we include in T^* the edges needed to ensure connectivity; they will be necessarily of the form $\{r, x^0\}$. Note that no demands will flow through these edges, so the capacity is always satisfied.

It is easy to observe that the topology T^* is a spanning tree, and that the demands flowing through its edges verify the capacities. Therefore T^* is such that the answer to the TE-MSTPDP is "yes".

Consider now, a spanning tree T^*, such that it satisfies the TE-MSTPDP.

For each variable $x \in X$, only one edge of $\{x^0, x^T\}$ or $\{x^0, x^F\}$ belongs in T^*. This owes to $\{x^T, x^F\}$ naturally being in T^*, as it is the only one through which we can send the demand between those two nodes, such that that capacity is met; and T^* being a spanning tree - if all three edges were to be T^*, we would have a cycle.

Therefore, we can construct a truth assignment, by assigning to each variable

$x \in X$ the value *true* if $\{x^0, x^T\} \in T^*$, the value *false* if $\{x^0, x^F\} \in T^*$, and an arbitrary value if $\{x^0, x^T\}, \{x^0, x^F\} \notin T^*$.

Moreover, there is an unique path between the root node r and the clause node $c \in C$. This path is necessarily of the form $\{r, x^0, x^T, c\}$ or $\{r, x^0, x^F, c\}$. Let us consider the absurd case where there exists a path p between r and c for which this is not true. The first possibility is that there exists another clause node $c' \in C$ on the path p, *e.g.* $p = \{r, x^0, x^{T'}, c', x^T, c\}$. Note however, that the demands both from r to c, and from r to c' ($d(r,c) + d(r,c') = 2$), flow through edge $\{x^{T'}, c'\}$, exceeding the capacity ($C_{\{x^{T'}, c'\}} = 1$). The same justification holds for the cases where there is more than one clause on path p. The second possibility is that there exists an edge of the type $\{x^T, x^F\}$ on path p, *e.g.* $\{r, x^0, x^T, x^F, c\}$. Nonetheless, the demands both from x^T to x^F and r to c ($d(x^T, x^F) + d(r, c') = \mu + 2$), flow through edge $\{x^T, x^F\}$ exceeding the capacity ($C_{\{x^T, x^F\}} = \mu + 1$).

Thus, as all the clauses are satisfied, T^* also describes a solution for SAT. Therefore, the TE-MSTPDP is polynomial reducible to SAT.

□

Corollary 1. *TE-MSTPP is $\mathcal{NP}$-hard.*

Proof. Consider the TE-MSTPDP(β), defined as a TE-MSTPDP where all the capacities are divided by β. Solving the TE-MSTPP is equivalent to solving a succession of TE-MSTPDP(β), with β given by a binary search algorithm, that iteratively updates either the upper bound for the optimal solution, if TE-MSTPDP(β) is feasible, or the lower bound, otherwise. The algorithm stops when the difference between the two is not larger than the desired precision. Hence, as the TE-MSTPDP is $\mathcal{NP}$-complete, the TE-MSTPP is $\mathcal{NP}$-hard. □

6. Problem formulation

Problems dealing with single spanning trees have been widely studied along the years. In these works, several mathematical programming formulations have

been suggested (see, for instance, [27]), that can be extended/adapted to the case of multiple spanning trees. In this section, we describe three such formulations.

In order to facilitate the exposition of the models, we divide the TE-MSTPP into three sub-problems, for which we provide different models. We consider the following sub-problems:

Sub-problem 1: Designing spanning trees;

Sub-problem 2: Routing the traffic demands;

Sub-problem 3: Determining the maximum link utilization.

In the next three sections, we propose MILP formulations for each of these sub-problems. In Section 6.4 we present the objective function of this problem. Finally, in Section 6.5 we describe complete formulations for the TE-MSTPP, that are obtained by combining adequately the formulations proposed in Sections 6.1, 6.2 and 6.3.

6.1. Sub-problem 1: Designing spanning trees

As mentioned above, there are different ways to model spanning trees as a MILP. In this section, we present two different formulations that can be used to model multiple spanning trees. These formulations are extensions of models that have been proposed in the literature for the minimum spanning tree problem.

The set of variables that are used in the proposed formulations are defined below. The first two set of variables, z and x, are defined in the directed graph G', whereas variables w are defined on the undirected graph G.

- $z_a^{us} = 1$ if arc $a = (i, j) \in A$ is used in the unique path from root node $u \in V$ to node j, in VLAN $s \in S$; 0 otherwise;
- $x_a^{uvs} = 1$ if arc $a \in A$ is used in the unique path from node $u \in V$ to node $v \in V$, in VLAN $s \in S$; 0 otherwise;
- $w_e^s = 1$ if edge $e \in E$ is used in VLAN $s \in S$; 0 otherwise.

6.1.1. Rooted directed formulation

The first formulation can be obtained by extending the model originally suggested by Martin [28] (see also [29]) for the minimum spanning tree problem.

Given the set of variables $\{z_a^{us}, w_e^s\}$, any feasible solution must verify the following set of constraints:

$$\sum_{a \in \delta^-(j)} z_a^{us} = 1, \qquad u, j \in V : u \neq j, s \in S \tag{1a}$$

$$z_{ij}^{us} + z_{ji}^{us} = w_e^s, \qquad u \in V, e = \{i, j\} \in E, s \in S \tag{1b}$$

$$z_a^{us} \in \{0, 1\}, \qquad u \in V, a = (i, j) \in A : j \neq u, s \in S \tag{1c}$$

$$w_e^s \in \{0, 1\}, \qquad e \in E, s \in S \tag{1d}$$

In each VLAN $s \in S$, we design an arborescence rooted at each node $u \in V$, by defining an unique path between the root node and every other node. As such, constraints (1a) state that in each arborescence, every node with the exception of the root, has an indegree of 1. In constraint set (1b) we ensure that all arborescences, in each VLAN, use the same edges. These two sets of constraints, in conjunction with (1c-1d), define the structure of each VLAN as a spanning tree.

6.1.2. Multicommodity flow formulation

The second formulation is based on multicommodity flows, which have been used to model spanning trees in several problems (*e.g.* [20, 21]). Given the set of variables $\{x_a^{uvs}, w_e^s\}$, any feasible solution for the problem must verify the set

of constraints below.

$$\sum_{a \in \delta^+(u)} x_a^{uvs} = 1, \qquad u, v \in V : u < v, s \in S \quad (2a)$$

$$\sum_{a \in \delta^-(i)} x_a^{uvs} - \sum_{a \in \delta^+(i)} x_a^{uvs} = 0, \qquad u, v, i \in V : u < v, i \neq \{u, v\}, s \in S \quad (2b)$$

$$x_{ij}^{uvs} + x_{ji}^{uvs} \leq w_e^s, \qquad u, v \in V : u < v, e = \{i, j\} \in E, s \in S \quad (2c)$$

$$\sum_{e \in E} w_e^s = n - 1, \qquad s \in S \quad (2d)$$

$$x_{ij}^{uvs} \in \{0, 1\}, \qquad (i, j) \in A, u, v \in V : u < v, j \neq u, v \neq i, s \in S \quad (2e)$$

$$w_e^s \in \{0, 1\}, \qquad e \in E, s \in S \quad (2f)$$

The flow conservation constraints (2a) and (2b) define, for each VLAN $s \in S$, a path between every pair of nodes, $u, v \in V$, through which the traffic is routed. Constraints (2c) guarantee that, in each VLAN, the paths only make use of the links chosen in the according VLAN. In (2d), the number of links used in each VLAN is set to $n - 1$, so that they form a spanning tree.

Finally, constraint sets (2e) and (2f) define integrality of the variables in the model.

Balakrishnan *et al.* [30] have suggested a set of constraints for enhancing the linking constraints for a general network design problem, that can be easily extended for the multiple spanning trees, as seen in (3a-3b).

$$x_{ij}^{uvs} + x_{ji}^{uv's} \leq w_e^s, \qquad u, v, v' \in V : v \neq v', e = \{i, j\} \in E, s \in S \quad (3a)$$

$$x_{ij}^{uvs} + x_{ji}^{u'vs} \leq w_e^s, \qquad u, u', v \in V : u \neq u', e = \{i, j\} \in E, s \in S \quad (3b)$$

The first set of constraints, (3a), states that when edge $e = \{i, j\}$ is used in a given VLAN s, then all traffic originated in a given node u, will flow either from i to j, or from j to i. Inequalities (3b) describes an equivalent situation, for all traffic flowing to a given node v.

In the following specific cases, (3a) and (3b) can be written as equalities,

instead of inequalities:

$$x_{ij}^{ujs} + x_{ji}^{uis} = w_e^s, \qquad u \in V, e = \{i,j\} \in E : u < j, u < i, s \in S \tag{4a}$$

$$x_{ij}^{ivs} + x_{ji}^{jvs} = w_e^s, \qquad v \in V, e = \{i,j\} \in E : i < v, j < v, s \in S \tag{4b}$$

Constraints (4a) assert that if edge $e = \{i,j\}$ is used in VLAN s, this necessarily means that either the traffic flowing between a given node u and j travels from i to j, or the traffic flowing between u and i travels in the opposite direction. Constraints (4b) describes an equivalent situation, where the common node between the two traffic demands is not the origin, but the destination.

Empirical evidence has revealed that for some instances, by using (3a - 3b) along with (4a - 4b), in the special cases previously mentioned, we are able to strengthen the bound of the LP relaxation of the previous model.

6.2. Sub-problem 2: Routing the traffic demands

Once the design for each VLAN is established, it is possible to route traffic flows according to the demands. To properly model this routing, we can look at the traffic demands in each VLAN in two distinct ways: either by considering the traffic demands between each pair of nodes separately, or by aggregating traffic that shares a single origin (or destination).

As seen in Section 6.1, the x variables describe a path between each pair of nodes. Therefore, it is natural to associate the first above-mentioned strategy with the use of these variables, as they allow for the immediate calculation of the quantity of traffic flowing through each edge, in each VLAN: $\sum_{\{u,v\}:u<v} d_s(u,v)\,(x_{ij}^{uvs} + x_{ji}^{uvs})$. In this sense, to model this second sub-problem, we need to define, for every VLAN, the unique path between each pair of nodes:

$$\sum_{a \in \delta^+(u)} x_a^{uvs} = 1, \qquad u, v \in V : u < v, s \in S \tag{5a}$$

$$\sum_{a \in \delta^-(i)} x_a^{uvs} - \sum_{a \in \delta^+(i)} x_a^{uvs} = 0, \qquad u, v, i \in V : u < v, i \neq \{u,v\}, s \in S \tag{5b}$$

$$x_{ij}^{uvs} \in \{0,1\}, \qquad (i,j) \in A, u, v \in V : u < v, j \neq u, v \neq i, s \in S \tag{5c}$$

Note that constraints (5a-5c) are the same as (2a-2b,2e). We repeat them here, for the sake of completeness, since they are also used to model the routing of the traffic demands. Observe, however, that when the we combine both, (5a-5c) are rendered redundant and are, therefore, omitted.

For the second strategy, in order to be able to aggregate traffic that have a common origin, the following variable set is defined:

- f_a^{us} = traffic quantity, originated from node $u \in V$, going through arc $a \in A$, in VLAN $s \in S$.

These variables are related to the x variables, since $f_{ij}^{us} = \sum_{v \in V \setminus \{u,i\}} d_s(u,v) \cdot x_{ij}^{uvs}$. They must verify the following sets of constraints, so that the traffic flows are correctly distributed:

$$\sum_{a \in \delta^+(u)} f_a^{us} = \sum_{v \in V : v \neq u} d_s(u,v), \qquad u \in V\colon u, s \in S \qquad \text{(6a)}$$

$$\sum_{a \in \delta^-(i)} f_a^{us} - \sum_{a \in \delta^+(i)} f_a^{us} = d_s(u,i), \qquad u, i \in V : u \neq i, s \in S \qquad \text{(6b)}$$

$$f_a^{us} \geq 0, \qquad u \in V, a = (i,j) \in A : j \neq u, s \in S \qquad \text{(6c)}$$

Constraint set (6a) defines the quantity of aggregated traffic flow leaving from each origin, to be sent to all other nodes. Constraints (6b) state that the difference between the traffic quantity originated from node u, in VLAN s, entering and exiting a given node i (different from u) must match, exactly, the traffic demand between node u and i, on that same VLAN.

6.3. Sub-problem 3: Link utilization and capacity constraints

The last sub-problem to consider deals, in fact, with two different components. Firstly, it guarantees that the distribution of the traffic flows, that was made in the second sub-problem, does not exceed the capacity of each link. Secondly, it calculates the utilization of each link. Nevertheless, as both these components can be dealt within the same constraint set, we see it as one sub-problem.

In order to calculate the maximum link utilization on the network, the following variable is defined:

- U^{max} = maximum value of link utilization.

As mentioned in the last section, the load on a link (total sum of traffic travelling through a link) can be calculated via either the x variables or f variables. Hence, there are two distinct constraint sets which can be used, depending on which variables are used: (7a) or (7b).

$$\sum_{s \in S} \sum_{u \in V} (f_{ij}^{us} + f_{ji}^{us}) \leq C_e \cdot U^{max}, \qquad e = \{i, j\} \in E \tag{7a}$$

or

$$\sum_{s \in S} \sum_{\{u,v\}: u<v} d_s(u, v)(x_{ij}^{uvs} + x_{ji}^{uvs}) \leq C_e \cdot U^{max}, \qquad e = \{i, j\} \in E \tag{7b}$$

$$0 \leq U^{max} \leq 1 \tag{7c}$$

On both (7a) and (7b), on the left hand side of each constraint the load on the link is calculated. That way, it is possible to determine, as well, the utilization of each link. As this is done for every existing link on the network, the variable U^{max} is attributed the value of the maximum utilization. At the same time, (7a) and (7b) bound the traffic quantity flowing through each link to the given capacity, as $U^{max} \in [0, 1]$ (7c).

6.4. Objective function

The objective of the problem is to minimize the maximum link utilization. As it was seen in the last section, every formulation uses variable U^{max}. Hence, their common objective function is the following:

$$\min \quad U^{max} \tag{8}$$

6.5. Complete formulations

Having modelled each one of the three sub-problems, it is now possible to go back and look at the original problem as a whole. We do this by combining adequately the formulations for the different sub-problems, which were presented

in the previous sections. The result are three complete formulations: the rooted directed formulation (RDF), the multicommodity flow formulation (MFF) and the rooted directed multicommodity flow formulation (RDMFF).

The RDF uses constraint sets (1a-1d) to define each VLAN as spanning tree and (6a-6c) to route the traffic demands. To link the variables involved in the formulations of these two sub-problems, we add the following constraints:

$$f_a^{us} \leq \sum_{v \in V \setminus \{u,i\}} d_s(u,v) \cdot z_a^{us}, \qquad i,j,u \in V : u \neq j, a \in A, s \in S \tag{9}$$

These constraints state that the traffic demands must flow through the arcs chosen for the respective VLAN.

Finally, the RDF uses (7a,7c) to calculate link utilization, whilst ensuring link capacity.

The MFF models the first sub-problem via (2a-2f). To route the traffic demands, the MFF uses (5a-5c). Notwithstanding, as mentioned before, these are omitted in the complete formulation, as they are the same as (2a-2b,2e). Finally to model the third sub-problem, the MFF uses (7b-7c).

It is also possible to combine the formulations proposed for Sub-problem 1 and 2 in a different way: we can opt to model the spanning trees as arborescences with (1a,1d), but use the multicommodity flows defined in (5a-5c) to route the traffic demands instead. The variables used for these two sub-problems are linked via the following constraints:

$$x_a^{uvs} \leq z_a^{us}, \qquad a \in A, u,v \in V : u < v, j \neq u, i \neq u, s \in S \tag{10}$$

These constraints imply that the traffic demands directed from node u to v can only flow through a given arc, if that arc is selected in the arborescence rooted at node u

This third complete formulation is named RDMFF. Empirical evidence has revealed that the bounds of LP relaxation of this formulation can be improved for some instances, by adding to it the linking valid inequalities below. They state that if arc (i,j) is on the path between node u and v, then the inverse

(j, i) is on the path between the root of the arborescence v, and node i.

$$x_{ij}^{uvs} \leq z_{ji}^{vs}, \qquad (i,j) \in A, u, v \in V : u < v, i \neq v, s \in S \tag{11}$$

For each model, listed in the first column, Table 1 describes the variable set, and enumerates which sets of constraints are used in the model to solve each sub-problem (*SP1, SP2, SP3*), as well as the sets of constraints used to link the first two sub-problems (*Link*). For the MFF and the RDMFF, it is also emphasized, in grey, the constraints needed to obtain their strengthened version: respectively, MFF+ and RDMFF+.

Model	Variables	SP1	SP2	Link	SP3
RDF	$\{U^{max}, f_a^{us}, z_a^{us}, w_e^s\}$	(1a-1d)	(6a-6c)	(9)	(7a,7c)
MFF*(+)*	$\{U^{max}, x_a^{uvs}, w_e^s\}$	(2a-2f) *(3a-3b)* *(4a-4b)*	(5a-5c)	-	(7b,7c)
RDMFF*(+)*	$\{U^{max}, x_a^{uvs}, z_a^{us}, w_e^s\}$	(1a-1d)	(5a-5c)	(10) *(11)*	(7b,7c)

Table 1: Composition of each complete formulation.

7. Comparing the LP relaxations

Consider the LP relaxations of the models introduced in the last section. In this section, we compare the strength of the LP relaxed models. An introduction of some of the concepts used in this section, can be found in Section 2.

Let $\mathcal{P}_{RD}$, $\mathcal{P}_{MF}$ and $\mathcal{P}_{RDMF}$ be the polyhedron defined by the set of feasible solutions of the LP relaxation of the RDF, MFF+ and RDMFF+, respectively.

Theorem 2. $Proj_{U^{max},w}(\mathcal{P}_{MF}) \subseteq Proj_{U^{max},w}(\mathcal{P}_{RD})$.

Proof. Theorem 2 can be proven by showing that any solution of the projection of $\mathcal{P}_{MF}$ onto the space of U^{max}, z, f and w, can be transformed to a solution in $\mathcal{P}_{RD}$. That is to say, any feasible solution of the LP relaxation of the MFF+ can

be converted to a solution that verifies all the constraints of the LP relaxation of the RDF.

Consider a generic solution of $\mathcal{P}_{MF}$, $\tilde{\mathcal{S}} = \{\tilde{w}_e^s, \tilde{x}_a^{uvs}, \tilde{U}^{max}\}$. Consider, as well, $\hat{\mathcal{S}} = \{\tilde{w}_e^s, \hat{z}_a^{us}, \hat{f}_a^{us}, \tilde{U}^{max}\}$, such that:

- $\hat{z}_a^{us} := \tilde{x}_a^{ujs} \qquad , u \in V, a = (i,j) \in A : j \neq u, s \in S$
- $\hat{f}_a^{us} := \sum_{v \in V \setminus \{u,i\}} d_s(u,v) \cdot \tilde{x}_a^{uvs} \qquad , a = (i,j) \in A : j \neq u, \forall s \in S$

The w and U^{max} variables, having the same meaning in both models, can be directly converted. The values of the z variables can be deduced from the $\tilde{x}$ variables in $\mathcal{P}_{MF}$. Finally, given the unique paths between each pair of nodes in a VLAN, it is easy to compute the quantity of traffic flowing each link. Thus, in a given VLAN, the traffic quantity originated in node u and flowing through arc a, is equal to the sum of traffic demands with origin in node u and destination in every other node v, such that the path between these two nodes goes through arc a. From this sum we exclude $v = u$ and $v = i$, as the x variables are not defined for these indexes' values.

In the following paragraphs, we demonstrate that $\hat{\mathcal{S}}$ satisfies all the constraints in the LP relaxation of the RDF.

Using (2a) and (2b), we can show that $\hat{\mathcal{S}}$ verifies constraint set (1a), for all $u, j \in V : u \neq j, s \in S$:

$$\sum_{a \in \delta^-(j)} \hat{z}_a^{us} = \sum_{a \in \delta^-(j)} \tilde{x}_a^{ujs} \overset{(2a+2b)}{=} 1$$

Through (4a), it can be seen that $\hat{\mathcal{S}}$ satisfies constraint set (1b), for all $u \in V, e = \{i,j\} \in E, s \in S$:

$$\hat{z}_{ij}^{us} + \hat{z}_{ji}^{us} = \tilde{x}_{ij}^{ujs} + \tilde{x}_{ji}^{uis} \overset{(4a)}{=} \tilde{w}_e^s$$

We can show that $\hat{\mathcal{S}}$ verifies constraint set (6a), for all $u \in V : u, s \in S$, via (2a).

$$\sum_{a\in\delta^+(u)} \hat{f}_a^{us} = \sum_{a=(i,j)\in\delta^+(u)} \sum_{v\in V\setminus\{u,i\}} d_s(u,v)\cdot\tilde{x}_a^{uvs}$$
$$= \sum_{v\in V\setminus u} d_s(u,v) \sum_{a\in\delta^+(u)} \tilde{x}_a^{uvs} \overset{(2a)}{=} \sum_{v\in V\setminus u} d_s(u,v)\cdot 1 = \sum_{v\in V\setminus u} d_s(u,v)$$

We can prove that $\hat{\mathcal{S}}$ verifies constraint set (6b), for all $u,i\in V : u\neq i, s\in S$, in the following way:

$$\sum_{a\in\delta^-(i)} \hat{f}_a^{us} - \sum_{a\in\delta^+(i)} \hat{f}_a^{us}$$
$$= \sum_{a\in\delta^-(i)} \sum_{v\in V\setminus\{u,j\}} d_s(u,v)\cdot\tilde{x}_a^{uvs} - \sum_{a\in\delta^+(i)} \sum_{v\in V\setminus\{u,i\}} d_s(u,v)\cdot\tilde{x}_a^{uvs}$$
$$= \sum_{v\in V\setminus u} d_s(u,v)\cdot\Big(\sum_{a\in\delta^-(i)} \tilde{x}_a^{uvs} - \sum_{a\in\delta^+(i)} \tilde{x}_a^{uvs}\Big)$$
$$= d_s(u,i)\cdot\Big(\sum_{a\in\delta^-(i)} \tilde{x}_a^{uis} - \sum_{a\in\delta^+(i)} \tilde{x}_a^{uis}\Big)$$
$$+ \sum_{v\in V\setminus\{u,i\}} d_s(u,v)\cdot\Big(\sum_{a\in\delta^-(i)} \tilde{x}_a^{uvs} - \sum_{a\in\delta^+(i)} \tilde{x}_a^{uvs}\Big)$$
$$\overset{(2a+2b)}{=} d_s(u,i)\cdot 1 + \sum_{v\in V\setminus\{u,i\}} d_s(u,v)\cdot 0 = d_s(u,i)$$

To prove that $\hat{\mathcal{S}}$ verifies constraints (9), for all $i,j,u\in V : u\neq j, a=(i,j)\in A, s\in S$, it is first necessary to show that $\tilde{x}_a^{uvs}\leq\tilde{x}_a^{ujs}$ for all cases. This can be achieved by replacing $\tilde{w}^s_{\{i,j\}}$, in the particular case of (3a) where $v'=i$, by the value given in (4a). The proof is completed as described below:

$$\hat{f}_a^{us} = \sum_{v\in V\setminus\{u,i\}} d_s(u,v)\cdot\tilde{x}_a^{uvs} \overset{(3a+4a)}{\leq} \sum_{v\in V\setminus\{u,i\}} d_s(u,v)\cdot\tilde{x}_a^{ujs} = \sum_{v\in V\setminus\{u,i\}} d_s(u,v)\cdot \hat{z}_a^{us}$$

As $\tilde{\mathcal{S}}$ verifies (7b), it is possible to prove that $\hat{\mathcal{S}}$ satisfies (7a), for all $e=\{i,j\}\in E$.

$$\sum_{s\in S}\sum_{u\in V}(\hat{f}_{ij}^{us}+\hat{f}_{ji}^{us}) = \sum_{s\in S}\sum_{u\in V}\sum_{v\in V\setminus\{u,i\}} d_s(u,v)\cdot(\tilde{x}_{ij}^{uvs}+\tilde{x}_{ji}^{uvs}) \overset{(7b)}{\leq} C_e\cdot\tilde{U}^{max}$$

The proof that $\mathcal{S}$ verifies the linear relaxation of constraint set (1c) and (1d), for every cases, is considered obvious.

□

Theorem 3. $Proj_{U^{max},x,w}(\mathcal{P}_{RDMF}) \subseteq \mathcal{P}_{MF}$.

Proof. Following the idea used to prove Theorem 2, Theorem 3 can be proved by showing that any feasible solution of the LP relaxation of the RDMFF+ has a corresponding solution in the LP relaxation of the MFF+.

Consider a generic solution on $\mathcal{P}_{RDMF}$, $\check{\mathcal{S}} = \{\check{w}_e^s, \check{x}_a^{uvs}, \check{U}^{max}, \check{z}_a^{us}\}$. We now show that the solution $\tilde{\mathcal{S}} = \{\check{w}_e^s, \check{x}_a^{uvs}, \check{U}^{max}\}$, belongs to $\mathcal{P}_{MF}$.

As the RDMFF+ uses the same variables as the MFF+, plus variable z, the transformation of $\check{\mathcal{S}}$ to $\tilde{\mathcal{S}}$ can be made directly. As most constraint sets of the RDMFF+ are inherited from the MFF+ the proof is obvious for most of them, namely (2a-2b), (2e-2f) and (7b-7c). Thus, we consider that it is only necessary to describe the proof that $\tilde{\mathcal{S}}$ respects constraint sets (2c), (2d), and the valid inequalities (3a-3b) and (4a-4b).

Using (10) and (1b), it is easy to show that $\mathcal{S}^*$ verifies (2c), for all $u, v \in V : u < v, e = \{i, j\} \in E, s \in S$.

$$\check{x}_{ij}^{uvs} + \check{x}_{ji}^{uvs} \overset{(10)}{\leq} \check{z}_{ij}^{us} + \check{z}_{ji}^{us} \overset{(1b)}{=} \check{w}_e^s$$

In order to prove that $\tilde{\mathcal{S}}$ satisfies constraint set (2d), for all $s \in S$, we use (1b) and (1a).

$$\begin{aligned}\sum_{e \in E} \check{w}_e^s &\overset{(1b)}{=} \sum_{e=\{i,j\} \in E} (\check{z}_{ij}^{us} + \check{z}_{ji}^{us}) \\ &= \frac{1}{2} \cdot \Big(\sum_{(i,j) \in A} \check{z}_{ij}^{us} + \sum_{(j,i) \in A} \check{z}_{ji}^{us} \Big) = \frac{1}{2} \cdot \Big(\sum_{j \in V} \sum_{a \in \delta^-(j)} \check{z}_a^{us} + \sum_{i \in V} \sum_{a \in \delta^-(i)} \check{z}_a^{us} \Big) \\ &\overset{(1a)}{=} \frac{1}{2} \cdot \Big(\sum_{j \in V : j \neq u} 1 + \sum_{i \in V : i \neq u} 1 \Big) = n - 1\end{aligned}$$

Via constraints (10) and (11), we can see that $\tilde{\mathcal{S}}$ verifies (3a), for all $u, v, v' \in V : v \neq v', e = \{i, j\} \in E, s \in S$. The proof that $\tilde{\mathcal{S}}$ verifies (3b) can achieved in the same fashion.

$$\check{x}_{ij}^{uvs} + \check{x}_{ji}^{uv's} \overset{(10+11)}{\leq} \check{z}_{ij}^{us} + \check{z}_{ji}^{us} \overset{(1b)}{=} \check{w}_e^s$$

In order to prove that $\tilde{\mathcal{S}}$ verifies (4a), we begin by demonstrating that $\check{x}_a^{ujs} = \check{z}_a^{us}$ for all $a \in A, u \in N : u < j, s \in S$. It is already known that $\check{x}_a^{ujs} \leq \check{z}_a^{us}$, by

(10). Hence, it is necessary to show that $\check{x}_a^{ujs} \geq \check{z}_a^{us}$:

$\check{x}_a^{ujs} \overset{(2a+2b)}{=} 1 - \sum_{(i',j)\in A: i' \neq i} \check{x}_{i'j}^{ujs} \overset{(10)}{\geq} 1 - \sum_{(i',j)\in A: i' \neq i} \check{z}_{i'j}^{us} \overset{(1a)}{=} \check{z}_a^{us}$

It now becomes easy to prove that $\tilde{\mathcal{S}}$ verifies (4a), for all $u \in V, e = \{i,j\} \in E : u < j, u < i, s \in S$:

$\check{x}_{ij}^{ujs} + \check{x}_{ji}^{uis} = \check{z}_{ij}^{us} + \check{z}_{ji}^{us} \overset{(1b)}{=} \check{w}_e^s$

The proof that $\tilde{\mathcal{S}}$ verifies (4b) can be achieved in a similar way.

□

As a consequence of Theorems 2 and 3, the LP relaxation of the RDMFF+ is also as strong as the LP relaxation of the RDF.

Corollary 2. $\mathcal{P}_{RDMF} \subseteq \mathcal{P}_{RD}$

A natural consequence of Theorems 2 and 3 is that $B_{LP}(RDMFF+) \geq B_{LP}(MFF+) \geq B_{LP}(RDF)$. Our computation experiments (see the next section) will reveal that for some instances, the LP bounds obtained can actually be different, in the sense that $B_{LP}(RDMFF+) > B_{LP}(MFF+)$, $B_{LP}(RDMFF+) > B_{LP}(RDF)$ or $B_{LP}(MFF+) > B_{LP}(RDF)$.

8. Computational experiments

In this section, we present the results of computational experiments, that can help to further evaluate the quality of the proposed formulations to solve the TE-MSTPP. These experiments were conducted using two distinct test sets of randomly generated instances, which aim at emulating two different types of network topologies. Instances in both test sets were created such that they span different values for the number of nodes and number of VLANs.

In the first test set, *t_rand*, the set of available links, whose number is calculated according to a given network density, was randomly distributed in the network. Each link was given a traffic capacity value of either 50, 75 or 100 Mbps. For each VLAN, the traffic demands (in Mbps) between nodes were generated following the formula proposed in [31]:

$$d_s(u,v) = \alpha(O_u D_v + O_v D_u) R_{(u,v)} e^{\frac{-L_2(u,v)}{2\Delta}}$$

For each node u, two random numbers, O_u and D_v are randomly generated in the interval $[0, 1]$. These values reflect, respectively, the attractiveness of each node as a sender and as a receiver. Another value, $R_{(u,v)}$, is generated in the same interval, for each pair of nodes. Parameter α is given as input. In these tests, the Euclidian distance (L_2) was substituted by the length of the shortest path between each pair of nodes, with respect to the number of links. Δ is the largest distance in the network. The final values were rounded to the nearest integer.

The second test set, *t_3tc*, follows the 3-Tier Cisco architecture, which is a common network topology in private enterprise data centers [32, 10]. This hierarchical architecture consists in a core, an aggregation and an edge tier. The core, at the top of the hierarchy, provides a gateway to the data center, from the extranet, WAN, or Internet edge. The switches in the second tier, serve as a bridge between the core and the nodes in the edge tier, aggregating the in and outflows. At the lowest level, the edge tier consists of racks of servers, interconnected by a Top of Rack (ToR) switch. We mimic this topology by generating a tripartite graph, in which around 1% of the nodes belong to the set representing the core tier, around 15% belong to the set representing the aggregation tier, and the remaining nodes stand for the ToR switches, in the edge tier. For each ToR we assign between 20 and 80 servers. As each server is only connected to the respective ToR switch, it is not necessary to represent them in the graph. Nevertheless, they are relevant for the generation of the traffic demands. Each ToR has 2 to 8 uplinks, depending on the size of the network. Each core node is connected to every node in the aggregation tier. Every link has a capacity value of 10 Gbps. For each VLAN, the traffic demands between the core nodes and the servers, or between servers, were calculated using the formula described above. All the traffic demands directed at (originating from) servers belonging to a given rack, are considered to have as a destination (origin) the node representing the corresponding ToR switch.

For each class of instances, five instances were generated and tested. Table 2 describes the classes of instances tested in these experiments, in terms of number of nodes in the network ($\#nodes$), network density (for *t_rand* only), and number of VLANs in the network ($\#VLANs$). α was given a value of 0.1 for every instance in *t_rand* (with the exception of instances of classes 10-12,14 where $\alpha = 0.01$), and of 10 for every instance in *t_3tc*, so that there was a low chance of having unfeasible instances.

Class ID	#nodes	#VLANs	density
t_rand_1	8	3	0.4
t_rand_2	8	6	0.4
t_rand_3	8	3	0.6
t_rand_4	8	2	0.8
t_rand_5	8	3	0.8
t_rand_6	10	2	0.5
t_rand_7	10	4	0.5
t_rand_8	12	2	0.3
t_rand_9	12	4	0.3
t_rand_10	12	6	0.3
t_rand_11	12	8	0.3
t_rand_12	12	10	0.3
t_rand_13	12	2	0.5
t_rand_14	12	3	0.5
t_3tc_1	12	4	
t_3tc_2	12	7	
t_3tc_3	15	4	
t_3tc_4	15	7	
t_3tc_5	20	2	
t_3tc_6	20	4	

Table 2: Description of each class of instances.

All the tests were performed on a Intel Core i7 CPU 960 @ 3.20GHz (x8) with 12GB of memory with 64 bits, and running Ubuntu 14.04.2 LTS (GNU/Linux $3.2.0-26-$generic x86_64).

ins.	Opt			Feas			MIP-time (s)			LP-time (s)			Gap_{LP}(%)			Gap_0 (%)			Gap_{end} (%)			B&B Tree Nodes		
	RDF	MFF	RDMFF	RDF	MFF	RDMFF	RDF	MFF	RDMFF	RDF	MFF	RDMFF	RDF	MFF	RDMFF	RDF	MFF	RDMFF	RDF	MFF	RDMFF	RDF	MFF	RDMFF
t_rand_1	5	4	5	5	5	5	246/1225	724/3600	16/74	0.1/0.3	0.7/1.9	0.3/0.5	19/55	17/45	16/42	17/44	16/45	15/42	0/0	10/51	0/0	3590/17334	9849/44862	43/139
t_rand_2	4	4	4	5	5	5	37/139	877/3600	744/3600	0.2/0.3	1.3/2.8	0.6/0.9	8/25	5/12	5/12	5/12	5/12	5/12	2/12	8/38	2/12	9279/33941	14904/33645	2645/9225
t_rand_3	5	4	5	5	5	5	71/252	1581/3600	36/113	0.2/0.3	1.1/1.5	1.1/1.6	20/35	20/35	20/32	19/35	20/35	20/32	0/0	1/3	0/0	3146/11259	173809/683569	156/426
t_rand_4	5	2	5	5	5	5	327/1422	3031/3600	45/112	0.2/0.2	1.3/1.5	0.4/0.4	60/67	59/65	57/63	60/67	59/65	54/63	0/0	21/43	0/0	26519/120587	87918/169600	171/374
t_rand_5	5	1	5	5	5	5	1387/3299	3125/3600	594/1816	0.3/0.3	1.5/1.7	0.5/0.6	44/53	40/48	38/46	40/48	40/48	38/46	0/0	36/52	0/0	37914/108059	37087/60000	1962/5468
t_rand_6	4	1	5	5	5	5	835/3600	3583/3600	348/586	0.2/0.3	2.3/2.9	3.3/4.8	52/62	51/60	50/59	51/62	52/62	49/59	4/21	39/59	0/0	52441/240731	37961/45805	584/1081
t_rand_7	0	0	1	5	5	5	3600/3600	3600/3600	2912/3600	0.3/0.3	7.5/12.3	1.8/2.0	33/42	29/36	26/32	31/31	30/36	26/32	27/27	38/38	23/23	77606/289710	39811/125300	1755/2704
t_rand_8	3	2	3	5	5	5	1485/3600	2291/3600	1473/3600	0.2/0.4	9.4/25.9	2.7/4.7	42/74	41/72	39/71	41/41	41/73	39/71	27/27	36/36	24/24	10177/30677	17025/64597	272/598
t_rand_9	2	0	4	5	5	5	2376/3600	3600/3600	1155/3600	0.3/0.3	4.1/4.6	2.2/2.5	7/12	7/11	6/10	6/6	7/11	5/10	4/4	13/13	1/1	33914/55200	31668/49935	2132/4398
t_rand_10	4	3	5	5	5	5	773/3600	2250/3600	367/913	0.3/0.4	5.2/6.8	3.9/5.6	4/15	4/13	4/13	4/4	4/11	4/13	1/1	14/14	0/0	65746/304601	545/1640	467/1827
t_rand_11	2	1	2	5	4	5	2310/3600	3018/3600	2361/3600	0.4/0.5	14./24.4	8.6/11.6	6/12	6/12	5/10	5/5	6/12	5/10	6/6	41/41	7/7	28636/51243	77/180	2862/10976
t_rand_12	2	1	2	5	5	5	2161/3600	3212/3600	2214/3600	0.5/0.5	18./33.5	8.8/10.8	4/12	4/11	2/8	3/3	3/10	2/7	4/4	39/39	2/2	36067/134137	198/968	6388/24557
t_rand_13	0	0	0	5	5	5	3600/3600	3600/3600	3600/3600	0.5/0.8	12./17.7	21.6/47.2	59/74	56/69	55/67	57/57	57/69	55/67	47/47	70/70	49/49	28302/40920	12480/26146	821/1440
t_rand_14	1	0	0	5	4	5	3545/3600	3600/3600	3600/3600	0.4/0.4	46./133.3	4.2/4.7	57/70	55/68	54/67	56/56	55/67	54/67	42/42	86/86	45/45	36589/65400	57/260	519/705
t_3tc_1	5	5	5	5	5	5	1/1	2/7	1/2	0.2/0.2	0.8/1.1	1.8/2.3	1/3	1/3	1/3	1/3	1/3	1/3	0/0	0/0	0/0	60/193	60/533	14/46
t_3tc_2	5	5	5	5	5	5	1/1	4/14	2/4	0.2/0.2	2.1/3.8	1.3/2.7	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	13/67	105/440	47/156
t_3tc_3	4	4	4	5	5	5	723/3600	62/130	743/3600	0.3/0.3	11.2/16.4	21.4/38.4	1/5	1/5	1/5	1/5	1/5	1/5	1/5	2/12	1/5	10335/51285	7199/30288	4125/20484
t_3tc_4	5	5	5	5	5	5	4/9	764/3406	55/136	0.4/0.6	15.3/32.0	12.8/30.6	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	69/216	17476/81600	63/220
t_3tc_5	3	1	3	5	4	5	1698/3600	2897/3600	1837/3600	0.3/0.4	65.4/97.3	63.3/75.1	16/35	16/35	16/35	16/35	16/35	16/35	9/35	38/100	10/31	16117/45463	19856/63348	4052/15849
t_3tc_6	5	1	5	5	1	5	15/24	3046/3600	662/934	0.6/0.8	147.6.3/219.4	51.1/83.5	1/3	1/3	1/3	1/3	0/2	1/3	0/0	80/100	0/0	167/250	268/1338	75/138
t_3tc_7	3	2	4	5	5	5	1443/3600	2502/3600	1116/3600	0.7/0.9	162.8/217.9	100.9/133.9	2/8	2/8	2/8	2/8	2/8	2/8	4/16	11/31	2/8	8084/26660	11135/23113	750/3462
t_3tc_8	0	0	0	5	0	5	3600/3600	3600/3600	3600/3600	0.5/0.6	400.9/696.0	168.6/218.0	27/41	27/41	27/41	27/41	27/41	27/41	27/41	100/100	72/74	10169/14244	0/0	4/6

Table 3: Computational results for each class of instances in *t_rand* and *t_3tc*.

The MILP and LP for each instance were solved with ILOG CLEX 12.6, using all 8 threads of the processor. A time limit of one hour was set. The results are described in Table 3. For every performance criteria we present both the average and the maximum values observed over the 5 instances of each class, separated by "/". We use the following notation in the table:

Opt: number of instances solved to optimality (note: time limit was of 3600 seconds);

Feas: number of instances for which CPLEX found a feasible solution;

MIP-time: Solving time for MIP;

LP-time: Solving time for LP relaxation of instances;

$\mathbf{Gap_{LP}}$: gap between overall best-found solution and lower bound obtained by LP relaxation;

$\mathbf{Gap_0}$: gap between overall best-found solution and lower bound at the end of the root node;

$\mathbf{Gap_{end}}$: gap between model's upper bound and lower bound at the end of solving (note that it might happen that the Gap_{end} is bigger than the Gap_0; this is because the best upper bound of the respective model, might be worse than the overall best upper bound, considered for the Gap_0);

B&B Tree Nodes: = number of nodes in the branch-and-bound tree (note that in some cases, this number is very low, because the LP relaxation is already challenging, *e.g.* t_3tc_8)

In order to understand the relative efficiency of each model for solving the TE-MSTPP, the performance profile of the MILP solving time is depicted in Figure 3. This performance profile measures the percentage of instances (spanned over the y-axis) that are solved in under different values, detailed in the x-axis. The graph implies that even though using the RDF with CPLEX is faster at solving "easy" instances (in terms of solving time), for the more "demanding"

ones, using CPLEX with the RDMFF+ proves to be faster, being able to solve more instances of both test sets in the given time limit. Using CPLEX with the MFF+ is significantly slower than with the other two formulations, for both test sets.

Furthermore, we can observe that the instances of the test set *t_3tc* are relatively easier to solve, when compared to the ones of *t_rand*. This can be explained by the fact that, albeit having a larger number of nodes, they are sparser.

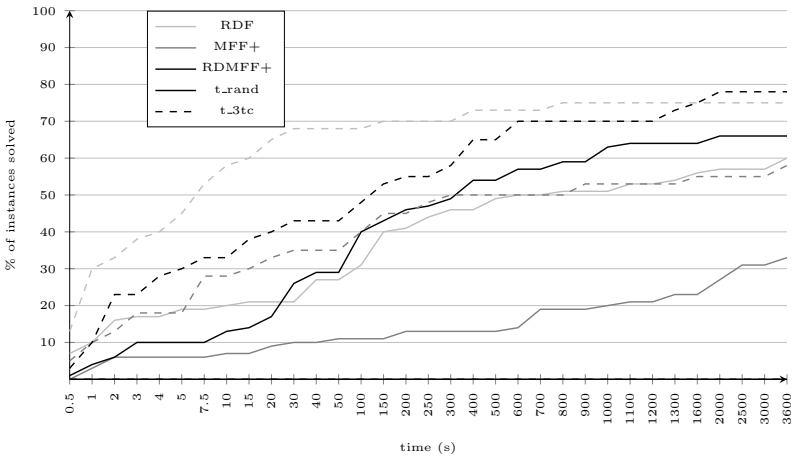


Figure 3: Performance profile of the solving time.

One other important criteria to consider, when evaluating the performance of each model in CPLEX, is the number of nodes in the branch-and-bound tree. Figure 4 presents the performance profile of each model, for instances that were solved within the time limit by every model. It illustrates the percentage of instances, whose branch-and-bound tree size is under the values described in the x-axis. We can observe that for instances of both test sets, the RDMFF+ seems to clearly favour a more compact branch-and-bound tree.

Finally, we looked into the lower bounds provided by each of the models. We consider two types of lower bounds: B_{LP}, the bound provided by solving the LP relaxation of the formulation; and B_0, the bound provided by CPLEX at the root node, after the solver added its own set of cuts. We name Gap_{LP} (Gap_0)

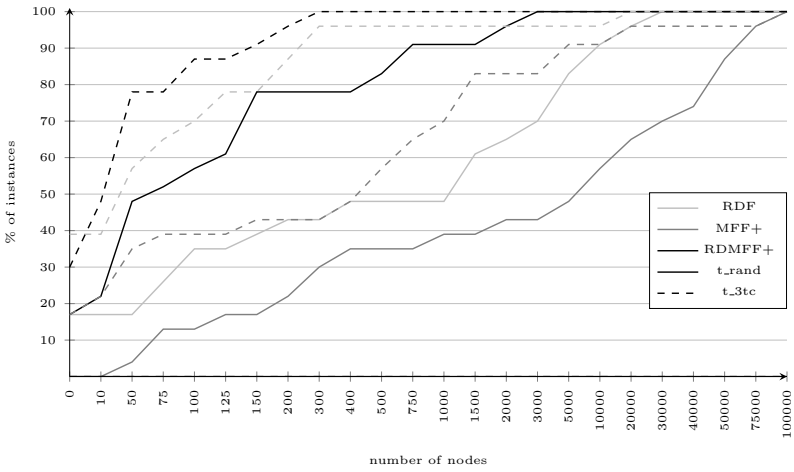


Figure 4: Performance profile of the number branch-and-bound tree nodes.

as the ratio between B_{LP} (B_0) and the optimum or best known value (U^{max*}). In order for the comparison between models to be valid, for each instance, both gaps were calculated taking into account the best integer solution found by any of the models, as seen in the following formula:

$Gap = \frac{U^{max*} - B}{U^{max*}}$

Figure 5 depicts the performance profile of the Gap_{LP} for each instance, by detailing how many instances provide gap values not bigger than the values in the x-axis. For test set *t_rand*, we can see what had been suggested at the end of Section 7: for some instances we see that $B_{LP}(RDMFF+) > B_{LP}(MFF+)$, $B_{LP}(RDMFF+) > B_{LP}(RDF)$ and/or $B_{LP}(MFF+) > B_{LP}(RDF)$.

Notwithstanding, the difference between the average gaps is not hugely significant: the average gap is 29.6% for the RDF, 28.2% for the MFF+ and 27.0% for the RDMFF+. This phenomenon is even more pronounced in the test set *t_3tc* where, for all instances, we have $B_{LP}(RDMFF+) = B_{LP}(MFF+) = B_{LP}(RDF)$, for all but one instance.

Moreover, an important aspect that can be observed in this computational experiments is the high fluctuation of these gap values, for instances of *t_rand*. For example, with the RDMFF+, the gap values varies between 0% and 71%.

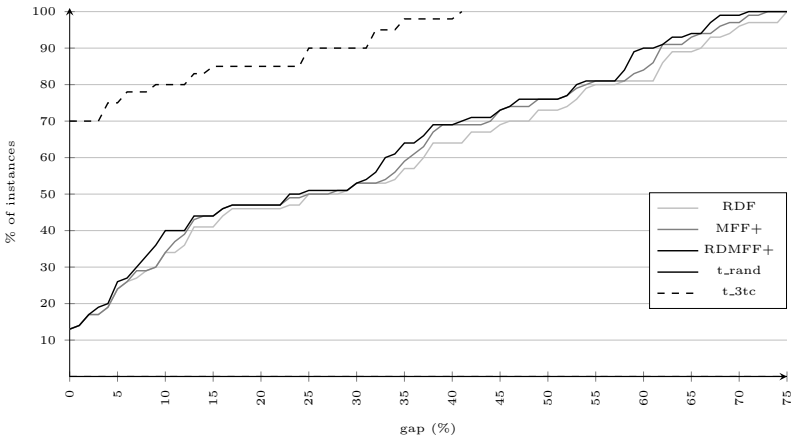


Figure 5: Performance profile of the Gap_{LP}.

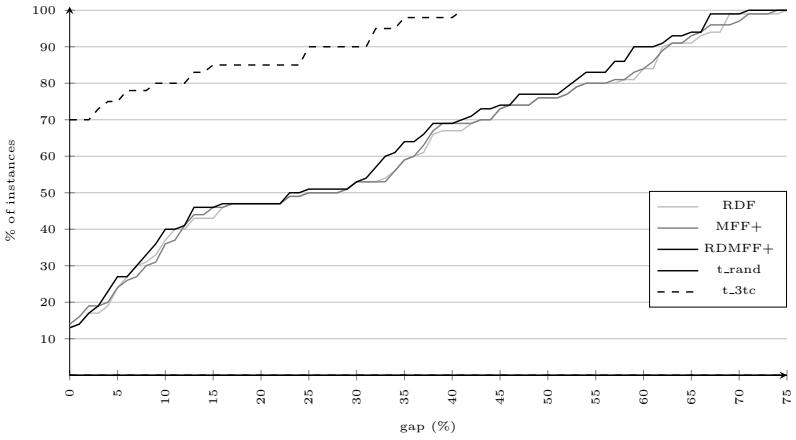


Figure 6: Performance profile of the Gap_0.

Even among each class for instances, which aggregate instances with the same basic characteristics, the gap values can be quite distinct, with the average standard deviation being of over 8%. Within test set *t_3tc*, this is not so evident, as for the majority of its instances, the gap value is 0%. The lower gap values verified with *t_3tc* might account for the equally lower computation times, seen back in Figure 3.

Another curious remark that can be made with respect to the test sets at hand is that, even though this gap tends to increase for bigger and/or denser

networks, as it would be expected, it tends to decrease when the number of VLANs increases.

Figure 5 describes the performance profile of the Gap_0. For test set *t_3tc* there are no improvements when comparing to the lower bound obtained by the LP relaxation. However, for test set *t_rand* we observe a slight decrease on the average gaps to 28.3% with the RDF, 28.1% with the MFF+, and 26.6% with the RDMFF+. Moreover, it is interesting to observe that, even if the trend is to have $B_0(RDMFF+) \geq B_0(MFF+) \geq B_0(RDF)$, as it was the case with the LP bounds, for some instances $B_0(MFF+) \geq B_0(RDMFF+)$.

9. Summary and conclusions

In this article, we studied the traffic engineering problem for the MSTP, that was first proposed in [10, 7] and that we denote as TE-MSTPP (see Section 3 for complete definition). We showed that this problem is $\mathcal{NP}$-hard.

We proposed three different MILP formulations: the RDF, the MFF and the RDMFF. We developed valid inequalities, that for some instances strengthen the bounds. We denote as MFF+ and RDMFF+, the combination of these formulations and the corresponding valid inequalities. We compared the strength of the LP relaxation of these models, and we showed that $B_{LP}(RDMFF+) \geq B_{LP}(MFF+) \geq B_{LP}(RDF)$.

Moreover, we presented a comprehensive array of computational experiments that were done, in order to further compare the proposed formulations. These experiments seem to point out the RDMFF+ as the most promising formulation, when used with CPLEX: the branch-and-bound trees are typically more compact; the gaps at the root node are, in most cases, smaller; and the more "difficult" instances are solved faster.

The results of the computational experiments also emphasize the difficulty of solving the TE-MSTPP optimally, even for relatively small instances. This is evidenced not only by the lengthy computation times, but also by the often weak LP relaxations. However, we stand by the importance of studies like the

one presented in this paper for the two reasons. First, the LP relaxations of MILP formulations provide lower bounds to the optimal solution, that allow us to evaluate the quality of the heuristic procedures existing in the literature. Second, the study of these models represents a crucial base for our future research, in which we will explore the utilization of higher-end mathematical programming methods, like Benders' decomposition in a branch-and-cut framework. We hope that this will allow us to tackle larger sized instances, thus narrowing the efficiency gap with respect to heuristic methods.

Acknowledgements

We would like to thank the two anonymous reviewers for their constructive comments that helped us to improve the paper and Amaro de Sousa for the useful discussion on the technology issues.

This work is supported by the Interuniversity Attraction Poles Programme P7/36 COMEX initiated by the Belgian Science Policy Office. This work was also partially supported by a grant from TCPER Nord-Pas de Calais/FEDER DATA Advanced data science and technologies 2015-2020. The work of Luís Gouveia is supported by National Funding from FCT - Fundação para a Ciência e a Tecnologia, under the project: UID/MAT/04561/201.

References

[1] V. B. Iversen, et al., Teletraffic engineering handbook, ITU-D SG 2 (2005) 16.

[2] P. E. Wirth, The role of teletraffic modeling in the new communications paradigms, Communications Magazine, IEEE 35 (8) (1997) 86–92.

[3] C. Kim, M. Caesar, J. Rexford, Floodless in Seattle: a scalable Ethernet architecture for large enterprises, in: ACM SIGCOMM Computer Communication Review, Vol. 38, ACM, 2008, pp. 3–14.

[4] IEEE Standard for Information Technology- Telecommunications and Information Exchange Between Systems- Local and Metropolitan Area Networks- Common Specifications Part 3: Media Access Control (MAC) Bridges, ANSI/IEEE Std 802.1D, 1998 Edition (1998) i–355.

[5] IEEE Standard for Local and Metropolitan Area Networks Virtual Bridged Local Area Networks, IEEE Std 802.1Q-2005 (Incorporates IEEE Std 802.1Q1998, IEEE Std 802.1u-2001, IEEE Std 802.1v-2001, and IEEE Std 802.1s-2002) (2006) 0_1–285.

[6] IEEE Standards for Local and Metropolitan Area Networks— Virtual Bridged Local Area Networks— Amendment 3: Multiple Spanning Trees, IEEE Std 802.1s-2002 (Amendment to IEEE Std 802.1Q, 1998 Edition) (2002) 0_1–211.

[7] T.-V. Ho, Traffic engineering techniques for data center networks, Ph.D. thesis, Ecole polytechnique de Louvain, Université catholique de Louvain (2012).

[8] A. Kolarov, B. Sengupta, A. Iwata, Design of multiple reverse spanning trees in next generation of ethernet-vpns, in: Global Telecommunications Conference, 2004. GLOBECOM'04. IEEE, Vol. 3, IEEE, 2004, pp. 1390–1395.

[9] A. Meddeb, Smart spanning tree bridging for carrier ethernets, in: Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE, IEEE, 2008, pp. 1–5.

[10] T.-V. Ho, Y. Deville, O. Bonaventure, P. Francois, Traffic engineering for multiple spanning tree protocol in large data centers, in: 23rd International Teletraffic Congress (ITC), IEEE, 2011, pp. 23–30.

[11] A. de Sousa, G. Soares, Improving Load Balance and Minimizing Service Disruption on Ethernet Networks with IEEE 802.1S MSTP, in: Workshop on IP QoS and Traffic Control, 2007, pp. 25–35.

[12] X. He, M. Zhu, Q. Chu, Traffic Engineering for Metro Ethernet Based on Multiple Spanning Trees, in: International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies (ICNICONSMCL'06), IEEE, 2006, pp. 97–97.

[13] Y. Lim, H. Yu, S. Das, S. S. Lee, M. Gerla, QoS-aware multiple spanning tree mechanism over a bridged LAN environment, in: GLOBECOM '03. IEEE Global Telecommunications Conference (IEEE Cat. No.03CH37489), Vol. 6, IEEE, 2003, pp. 3068–3072.

[14] A. Meddeb, Multiple Spanning Tree Generation and Mapping Algorithms for Carrier Class Ethernets, in: IEEE Globecom 2006, IEEE, 2006, pp. 1–5.

[15] D. Santos, A. de Sousa, F. Alvelos, M. Dzida, M. Pióro, Optimization of link load balancing in multiple spanning tree routing networks, Telecommunication Systems 48 (1-2) (2010) 109–124.

[16] D. Santos, A. de Sousa, F. Alvelos, M. Dzida, M. Pióro, M. Zagozdzon, Traffic Engineering of Multiple Spanning Tree Routing Networks : the Load Balancing Case, in: Next Generation Internet Networks, 2009. NGI '09, 2009, pp. 1–8.

[17] W. Chen, D. Jin, L. Zeng, Design of Multiple Spanning Trees for Traffic Engineering in Metro Ethernet, in: 2006 International Conference on Communication Technology, IEEE, 2006, pp. 1–4.

[18] M. Padmaraj, S. Nair, M. Marchetti, G. Chiruvolu, M. Ali, A. Ge, Metro Ethernet traffic engineering based on optimal multiple spanning trees˜, in: Second IFIP International Conference on Wireless and Optical Communications Networks, 2005. WOCN 2005., IEEE, 2005, pp. 568–572.

[19] G. Mirjalily, F. A. Sigari, R. Saadat, Best Multiple Spanning Tree in Metro Ethernet Networks, in: 2009 Second International Conference on Computer and Electrical Engineering, IEEE, 2009, pp. 117–121.

[20] T. Cinkler, A. Kern, I. Moldován, Optimized QoS protection of Ethernet trees, in: Design of Reliable Communication Networks, 2005.(DRCN 2005). Proceedings. 5th International Workshop on, IEEE, 2005, pp. 8–pp.

[21] T. Cinkler, I. Moldován, A. Kern, C. Lukovszki, G. Sallai, Optimizing QoS aware Ethernet spanning trees, MSAN 2005 1 (2005) 30–34.

[22] A. Capone, D. Corti, L. Gianoli, B. Sansó, An optimization framework for the energy management of carrier ethernet networks with multiple spanning trees, Computer Networks 56 (17) (2012) 3666–3681.

[23] S. S. Lee, K.-Y. Li, C.-C. Lin, Modeling and algorithm for multiple spanning tree provisioning in resilient and load balanced ethernet networks, Mathematical Problems in Engineering 2015.

[24] T. C. Hu, Optimum communication spanning trees, SIAM Journal on Computing 3 (3) (1974) 188–195.

[25] S. A. Cook, The complexity of theorem-proving procedures, in: Proceedings of the third annual ACM symposium on Theory of computing, ACM, 1971, pp. 151–158.

[26] L. A. Levin, Universal sequential search problems, Problemy Peredachi Informatsii 9 (3) (1973) 115–116.

[27] T. L. Magnanti, L. A. Wolsey, Optimal trees, Handbooks in operations research and management science 7 (1995) 503–615.

[28] R. K. Martin, Using separation algorithms to generate mixed integer model reformulations, Operations Research Letters 10 (3) (1991) 119–128.

[29] M. Conforti, G. Cornuéjols, G. Zambelli, Extended formulations in combinatorial optimization, 4OR 8 (1) (2010) 1–48.

[30] A. Balakrishnan, T. L. Magnanti, R. T. Wong, A dual-ascent procedure for large-scale uncapacitated network design, Operations Research 37 (5) (1989) 716–740.

[31] B. Fortz, M. Thorup, Increasing internet capacity using local search, Computational Optimization and Applications 29 (1) (2004) 13–48.

[32] C. D. C. Infrastructure, 2.5 design guide, cisco systems, Inc, San Jose, CA.